Classification vs Regression

Classification vs Regression

Which type of supervised machine learning problem is this, classification or regression? Why?

This is a classification problem because we are trying to predict if student will drop out or not based on available data

Exploring the Data

Data exploration

Student response addresses the most important characteristics of the dataset and uses these characteristics to inform their decision making. Important characteristics must include:

```
Total number of students: 395
Number of students who passed: 265
Number of students who failed: 130
Number of features: 30
Graduation rate of the class: 67.09%
```

Preparing the Data

Identify feature and target columns

Code has been executed in the iPython notebook, with proper output and no errors.

```
In [4]: # Extract feature (X) and target (y) columns
        feature_cols = list(student_data.columns[:-1])  # all columns but last are features
        target_col = student_data.columns[-1]  # last column is the target/label
        print "Feature column(s):-\n{}".format(feature_cols)
        print "Target column: {}".format(target_col)

        X_all = student_data[feature_cols]  # feature values for all students
        y_all = student_data[target_col]  # corresponding targets/labels
        print "\nFeature values:-"
        print X_all.head()  # print the first 5 rows

Feature column(s):-
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']
Target column: passed

Feature values:-
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob     Fjob  \
0     GP   F   18       U     GT3       A     4     4  at_home  teacher
1     GP   F   17       U     GT3       T     1     1  at_home    other
2     GP   F   15       U     LE3       T     1     1  at_home    other
3     GP   F   15       U     GT3       T     4     2   health services
4     GP   F   16       U     GT3       T     3     3    other    other

    ...    higher internet  romantic  famrel  freetime  goout  Dalc  Walc  \
0   ...       yes       no        no       4         3      4     1     1
1   ...       yes      yes        no       5         3      3     1     1
2   ...       yes      yes        no       4         3      2     2     3
3   ...       yes      yes       yes       3         2      2     1     1
4   ...       yes       no        no       4         3      2     1     2

  absences
0        6
1        4
2       10
3        2
4        4
```

Preprocess feature columns

Code has been executed in the iPython notebook, with proper output and no errors.

```
Preprocess feature columns

As you can see, there are several non-numeric columns that need to be converted! Many of them are simply yes/no, e.g. internet. These can be reasonably converted into 1/0 (binary) values.

Other columns, like Mjob and Fjob, have more than two values, and are known as categorical variables. The recommended way to handle such a column is to create as many columns as possible values (e.g. Fjob_teacher, Fjob_other, Fjob_services, etc.), and assign a 1 to one of them and 0 to all others.

These generated columns are sometimes called dummy variables, and we will use the pandas.get_dummies() function to perform this transformation.

In [5]: # Preprocess feature columns
        def preprocess_features(X):
            outX = pd.DataFrame(index=X.index)  # output dataframe, initially empty

            # Check each column
            for col, col_data in X.iteritems():
                # If data type is non-numeric, try to replace all yes/no values with 1/0
                if col_data.dtype == object:
                    col_data = col_data.replace(['yes', 'no'], [1, 0])
                # Note: This should change the data type for yes/no columns to int

                # If still non-numeric, convert to one or more dummy variables
                if col_data.dtype == object:
                    col_data = pd.get_dummies(col_data, prefix=col)  # e.g. 'school' => 'school_GP', 'school_MS'

                outX = outX.join(col_data)  # collect column(s) in output dataframe

            return outX

        X_all = preprocess_features(X_all)
        print "Processed feature columns ({}):-\n{}".format(len(X_all.columns), list(X_all.columns))

        Processed feature columns (48):-
        ['school_GP', 'school_MS', 'sex_F', 'sex_M', 'age', 'address_R', 'address_U', 'famsize_GT3', 'famsize_LE3', 'Pstatus_A', 'Pstatus_T', 'Medu', 'Fedu', 'Mjob_at_home', 'Mjob_health', 'Mjob_other', 'Mjob_services', 'Mjob_teacher', 'Fjob_at_home', 'Fjob_health', 'Fjob_other', 'Fjob_services', 'Fjob_teacher', 'reason_course', 'reason_home', 'reason_other', 'reason_reputation', 'guardian_father', 'guardian_mother', 'guardian_other', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']
```

Split data into training and test sets

Training and test sets have been generated by randomly sampling the overall dataset.

```
Split data into training and test sets

So far, we have converted all categorical features into numeric values. In this next step, we split the data (both features and corresponding labels) into training and test sets.

In [13]: from sklearn.cross_validation import train_test_split

         # TODO: Set the number of training points
         num_train = 300

         # Set the number of testing points
         num_test = X_all.shape[0] - num_train

         # TODO: Then, select features (X) and corresponding labels (y) for the training and test sets
         # Shuffle the data or randomly select samples to avoid any bias due to ordering in the dataset
         X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size = num_test, random_state= )

         print "Training set: {} samples".format(X_train.shape[0])
         print "Test set: {} samples".format(X_test.shape[0])
         # Note: If you need a validation set, extract it from within training data

         Training set: 300 samples
         Test set: 95 samples
```

Training and Evaluating Models

Time and Space Complexity

Both the big-O notation for the space complexity to represent the model and the time for the algorithm to make a prediction are provided, or a list of several of the major factors that affect the time & space complexities are presented with a description of the largest driving factor as constant, linear, logarithmic, polynomial, etc in nature. Student presents resources or reasonable justification for their response.

Due to small dataset the prediction and training times of the model are negleble. Driving factor of computation is liner in nature, meaning as we increase number of samples, the time it will take to execute will increase. As such driving factor is Linear.https://en.wikipedia.org/wiki/Time_complexity#Polynomial_time

Model Application

Decision Tree Classifier

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to explore.

- What are the general applications of this model?

Decision Tree Classifier applies a straitforward idea to solve the classification problem. it poses a series of carefully crafted questions about the attributes of the test record. Each time time it receive an answer, a follow-up question is asked until a conclusion about the calss label of the record is reached.

- What are its strengths and weaknesses?

Advantages: Decision trees are simple to use, easy to understand.
Disadvantages: Even a small change in input data can at times, cause large changes in the tree. Decision trees are also prone to errors in classification, owing to differences in perceptions and the limitations of applying statistical tools.

- Given what you know about the data so far, why did you choose this model to apply?

Predictive Power, Relative Simplicity

Model Application

KNeighborsClassifier

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to explore.

- What are the general applications of this model?

KNeighborsClassifier classifies data by evaluating classes and data point and classes of K nearest neighbors. Class is determined based on majority vote. In case of student data we only have classes and we measure distance to nearest neighbor through parameter search process we found that number to be 7). When the model encounters an example it has not see it will compare the distance of such example from neighbors around it. Model uses neighbors majority vote to determine class of the new example. Its important to use uneven number of neighbors to achieve tie breaks in majority voting.

- What are its strengths and weaknesses?

Advantages: Relatively fast as only distances are measured .

- **Robust to noisy training data**

Disadvantages: Initial placement of cluster centroid can affect clustering, we have to adjust centroids as we cluster. Even after this examples that are close together can switch cluster assignments.

- Need to determine value of parameter K (number of nearest neighbors)
- Given what you know about the data so far, why did you choose this model to apply?
  - I originally chose this model as one of the possible models because it does well generalizing data and selecting membership based on similarity. Meaning if new evaluation point is close or falls into an existing group then we can easily clarify it based on distance from other members of its class.

Model Application

SVM

The pros and cons or application for each model is provided with reasonable justification why each model was chosen to explore.

- What are the general applications of this model?

SVM classifies data by separating classes and than placing as wide of a margin as possible that will still achieve linear separation. We can use this model to linearly separate the data with little fear that outliers will cause imbalance.

- What are its strengths and weaknesses?
- **Advantage** SVMs have good generalization performance
- **Disadvantage:** limitation of the support vector approach lies in choice of the kernel Burgess (1998)
- Given what you know about the data so far, why did you choose this model to apply?
  - I originally chose this model as one of the possible models because it separates data with a margin allowing for classes of new data points to be determined with limited overfitting.

Model Performance Metrics

All the required time and F1 scores for each model and training set sizes are provided within the chart given. The performance metrics are reasonable relative to other models measured.

Student Intervention System - Decision Tree Classifier

	Training set size		
	100	200	300
Training time (secs)	0	0	0
Prediction time (secs)	0	0	0
F1 score for training set	0.905	0.830	0.809
F1 score for test set	0.802	0.794	0.795

Student Intervention System - LinearSVC Classifier

	Training set size		
	100	200	300
Training time (secs)	0.007	0.018	0.024
Prediction time (secs)	0	0	0
F1 score for training set	0.797	0.840	0.833
F1 score for test set	0.844	0.847	0.788

Student Intervention System - KNeighborsClassifier Classifier

	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.002	0.002	0.006
F1 score for training set	0.781	0.830	0.841
F1 score for test set	0.795	0.783	0.788

Choosing the Best Model

Choosing the Optimal Model

- Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model. Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?

Based on tests performed, I compared 3 models, Decision tree classifier, Random forest, and Support Vector Classification. After evaluating performance of each model I choose Decision Tree Classifier Model for number of reasons:

- Accuracy: Model predicts with high F1 Score. F1 score conveys the balance between the precision and the recall.

- Performance: Obtainings training and test prediction fastes

- In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a Decision Tree or Support Vector Machine, how does it make a prediction).

KNeighborsClassifier Classifier uses a majority neibour voting to detrimine the class of a new example

- Fine-tune the model. Use Gridsearch with at least one important parameter tuned and with at least 3 settings. Use the entire training set for this.
Without finding the best parameters following were the prediction accuracies:

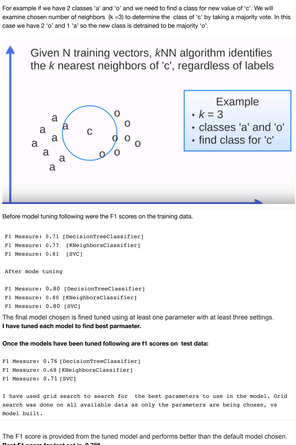
I have tuned each model to find best parmaeter. Please refer to outputs of each model

- What is the model's final F1 score?
Best F1 score for test set is 0.788

Describing the Model in Layman's Terms

KNeighborsClassifier Classifier uses a majority neibour voting to detrimine the class of a new example.

For example if we have 2 classes 'a' and 'c' and we need to find a class for new value of 'c'. We will examine chosen number of neighbors (k =3) to determine the class of 'c' by taking a majority vote. In this case we have 2 'c' and 1 'a' so the new class is detained to be majority 'c'.



Model Tuning

Before model tuning following were the F1 scores on the training data.

```
F1 Measures: 0.71 [DecisionTreeClassifier]
F1 Measures: 0.77  [KNeighborsClassifier]
F1 Measures: 0.81  [SVC]

After mode tuning

F1 Measures: 0.80 [DecisionTreeClassifier]
F1 Measures: 0.80 [KNeighborsClassifier]
F1 Measures: 0.80 [SVC]
```

The final model chosen is fined tuned using at least one parameter with at least three settings.
I have tuned each model to find best parmaeter.

Once the models have been tuned following are f1 scores on test data:

```
F1 Measures: 0.76 [DecisionTreeClassifier]
F1 Measures: 0.68 [KNeighborsClassifier]
F1 Measures: 0.71 [SVC]

I have used grid search to search for  the best parameters to use in the model. Grid search was done on all available data as only the parameters are being chosen, vs model built.
```

Tuned F1 Score

The F1 score is provided from the tuned model and performs better than the default model chosen.
Best F1 score for test set is 0.788

Quality of Code

Functionality

Code reflects the description in the documentation.